Decision Procedures for Multisets with Cardinality Constraints

Ruzica Piskac and Viktor Kuncak

School of Computer and Communication Science
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract. Applications in software verification and interactive theorem proving often involve reasoning about sets of objects. Cardinality constraints on such collections also arise in these scenarios. Multisets arise for analogous reasons as sets: abstracting the content of linked data structure with duplicate elements leads to multisets. Interactive theorem provers such as Isabelle specify theories of multisets and prove a number of theorems about them to enable their use in interactive verification. However, the decidability and complexity of constraints on multisets is much less understood than for constraints on sets.
The first contribution of this paper is a polynomial-space algorithm for deciding expressive quantifier-free constraints on multisets with cardinality operators. Our decision procedure reduces in polynomial time constraints on multisets to constraints in an extension of quantifier-free Presburger arithmetic with certain "unbounded sum" expressions. We prove bounds on solutions of resulting constraints and describe a polynomial-space decision procedure for these constraints.
The second contribution of this paper is a proof that adding quantifiers to a constraint language containing subset and cardinality operators yields undecidable constraints. The result follows by reduction from Hilbert's 10th problem.

This version differs from the published one in correcting the statement of the intermediate result, Theorem 3, which was originally stated in an unnecessarily general form. The final statement of the paper (Corollary 1) remains unaffected.

1 Introduction

Collections of objects are fundamental and ubiquitous concepts in computer science and mathematics. It is therefore not surprising that they often arise in software analysis and verification [1], as well as in interactive theorem proving [19]. Moreover, such constraints often involve cardinality bounds on collections. Recent work describes decision procedures for constraints that involve sets and their cardinalities [10, 12], characterizing the complexity of both quantified and quantifier-free constraints.

In many applications it is more appropriate to use multisets (bags) rather than sets as a way of representing collections of objects. It is therefore interesting to consider constraints on multisets along with cardinality bounds. There is a range of useful operations and relations on multisets, beyond the traditional disjoint union and difference. These operations are all definable using quantifier-free Presburger arithmetic (QFPA) formulas on the number of occurrences of each element in the multiset. This paper describes such a language that admits reasoning about integers, sets and multisets, supports standard set and multiset operations as well as any QFPA-definable operation on multisets (including the conversion of a multiset into a set), and supports a cardinality operator that counts the total number of elements. We present a decision procedure for this language, provide a PSPACE upper bound on the decision problem, and show that its extension with quantifiers is undecidable.

Our language can express sets as a special kind of multisets, so our new decision procedure is also a decision procedure for constraints in [12]. However, multisets introduce considerable additional difficulty compared to the original problem. For example, while the number of non-equivalent set terms is finite (they are unions of Venn regions), the number of non-equivalent multiset terms is infinite due to non-idempotent operators such as multiset sum $\uplus$. Instead of considering Venn regions, our new algorithm reduces multiset constraints to an extension of Presburger arithmetic with a particular sum expression. To decide this extension, we rely on the fact that solutions of formulas of Presburger arithmetic are semilinear sets [8] and use bounds on generators of such sets [23] to establish small model property for this extension.

Previously, Zarba [25] considered decision procedures for quantifier-free multisets but without the cardinality operator, showing that it reduces to quantifier-free pointwise reasoning. However, the cardinality operator makes that reduction impossible. More recently Lugiez [13] showed (in the context of a more general result on multitree automata) the decidability of quantified constraints with a weaker form of cardinality operator that counts only distinct elements in a multiset, and shows decidability of certain quantifier-free expressible constraints with cardinality operator. Regarding quantified constraints with the general cardinality operator, [13, Section 3.4] states "the status of the complete logic is still an open problem". We resolve this question, showing that the quantified constraints with cardinality are undecidable (Section 6). The decidable quantified constraints in [13] allow quantifiers that can be eliminated to obtain quantifier-free constraints, which can then be expressed using the decidable constraints in the present paper. We analyze the complexity of the decision problem for our quantifier-free constraints, and show that it belongs to PSPACE, which is the first complexity bound for constraints on multisets with a cardinality operator.

Contributions. We summarize our contributions as follows.

1. We show how to decide expressive quantifier-free constraints on multisets and cardinality operators in polynomial space, and
2. We show that adding quantifiers to a constraint language containing subset and cardinality operators yields undecidable constraints.

Overview. We continue by presenting examples that motivate the constraints we consider in this paper. We then outline our decision procedure through an example. The main part of the paper describes the decision procedure and its correctness. We then show that an extension of our constraints with quantifiers is undecidable.

1.1 Multisets in Interactive Verification

As an example of using multisets with cardinalities in interactive verification, consider the Multiset library [20] of the interactive theorem prover Isabelle [19]. This library represents a multiset as a function f from some (parameter) type to the set of natural numbers, such that the set S of elements x with $f(x) > 0$ is finite. It defines the `size` function on multisets as the sum of $f(x)$ over all $x \in S$. Several lemmas proved in the library itself mention both multisets and the `size` function, such as the `size_union` lemma ($\texttt{size}(M + N) = \texttt{size}\ M + \texttt{size}\ N$), where `+` on the left-hand side is resolved as the disjoint multiset union. Other Isabelle theories build on the Multiset library, including the Permutation library for reasoning about permutations, formalization [22] of the UNITY parallel programming approach [17], and example specifications of sorting algorithms.

This paper considers such a theory of multisets with size constraints. For simplicity, we fix the set E from which multiset elements are drawn. We assume E to be finite, but of unknown cardinality. If m is a multiset, we call $\texttt{size}(m)$ the cardinality of m, and denote it $|m|$ in this paper. As an example, the `size_union` lemma in our notation becomes $|M \uplus N| = |M| + |N|$.

1.2 Multisets in Software Analysis and Verification

It is often desirable to abstract the content of mutable and immutable data structures into collections to raise the level of abstraction when reasoning about programs. Abstracting linked structures as sets and relations enables high-level reasoning in verification systems such as Jahob [9]. For collections that may contain duplicates, abstraction using multisets is more precise than abstraction using sets. The decision procedure described in this paper would therefore enable reasoning about such precise abstractions, analogously to the way current decision procedures enable reasoning with set abstraction.

To illustrate the role of the cardinality operator, note that data structure implementations often contains integer size fields. If s is a data structure size field and L an abstract multiset field denoting data structure content, data structure operations need to preserve the invariant $s = |L|$. When verifying an insertion of an element x into a container, we therefore obtain verification conditions such as $|L|{=}s \rightarrow |L \uplus x|{=}s{+}1$. When verifying deletion of an element from a container we obtain verification conditions such as

$$x \subseteq L \wedge |x| = 1 \rightarrow |L \backslash x| = |L| - 1 \tag{1}$$

The decision procedure described in this paper can prove such verification conditions.

To describe data structure operations it is useful to have not only operations such as disjoint union $\uplus$ and set difference, but also an operation that, given multisets m_1 and m_2 produces a multiset m_0 which is the result of removing from m_1 *all* occurrences of elements that occur in m_2. Let $m_i(e)$ denote the number of occurrences of an element e in multiset m_i. Then we can specify such a removal operation with $\forall e.(m_2(e) = 0 \implies m_0(e) = m_1(e)) \wedge (m_2(e) > 0 \rightarrow m_0(e) = 0)$. We introduce a shorthand $m_0 = m_1 \setminus\!\setminus m_2$ for this formula. Our constraints support any such operation definable pointwise by QFPA formula.

Multisets have already been used in the verification system for data structures with bag and size properties [18], which invokes Isabelle to prove the generated multiset constraints. Our paper describes a decision procedure for a language of multisets with cardinalities, which could be used both within program verification systems and within interactive theorem provers, obtaining completeness and improved efficiency for a well-defined class of formulas. In addition to reasoning about abstractions of data structures, our constraints can be used directly to specify properties in programming languages such as SETL [24], which has built in set data type, and in the Gamma parallel programming paradigm [2, Page 103] based on multiset transformations.

2 Decision Procedure through an Example

We next outline our decision procedure by applying it informally to the constraint (1) from the previous section. This section demonstrates only the main idea of the algorithm; Sections 4 and 5 give the detailed description (the reader may wish to revisit this example after reading those details). To prove validity of (1), we show that its negation,

$$x \subseteq L \wedge |x| = 1 \wedge |L \setminus x| \neq |L| - 1, \tag{2}$$

is unsatisfiable. Our algorithm expresses a given formula through quantifier-free Presburger arithmetic (QFPA) extended with sum expressions $\sum t$ over other QFPA terms t. In such sum expressions, the number of occurrences of an element e in a multiset m is denoted by $m(e)$. Every sum ranges over all elements e of some fixed domain E of unknown size. For example, our algorithm converts $|x| = 1$ into $\sum x(e) = 1$.

We also allow conditional expressions ite in our formulas. If c is a QFPA formula and t_1 and t_2 two QFPA terms, then $\mathsf{ite}(c, t_1, t_2)$ has value t_1 when c holds, and t_2 otherwise. A multiset inclusion $x \subseteq L$ becomes $\forall e.\, x(e) \leq L(e)$ which in turn is transformed into the sum $\sum \mathsf{ite}(x(e) \leq L(e), 0, 1) = 0$. By introducing fresh variables x_1 for $L \setminus x$, we obtain formula $x_1 = L \setminus x$. The formula $x_1 = L \setminus x$ becomes the sum $\sum \mathsf{ite}(x_1(e) = \mathsf{ite}(L(e) \leq x(e), 0, L(e) - x(e)), 0, 1) = 0$. Formula (2) therefore becomes

$$\sum \mathsf{ite}(x(e) \leq L(e), 0, 1) = 0 \;\wedge\; \sum x(e) = 1 \;\wedge\; \sum x_1(e) \neq \sum L(e) - 1 \;\wedge \\ \sum \mathsf{ite}(x_1(e) = \mathsf{ite}(L(e) \leq x(e), 0, L(e) - x(e)), 0, 1) = 0 \tag{3}$$

Because every sum ranges over the same set of elements $e \in E$, we can combine all sums into one sum with vector summands. Introducing k_1 for $|x_1|$, and k_2 for $|L|$, we obtain the formula

$$k_1 \neq k_2 - 1 \ \wedge \ (k_1, k_2, 1, 0, 0) = \sum \Big(x_1(e), L(e), x(e), \\ \mathsf{ite}(x_1(e) = \mathsf{ite}(L(e) \leq x(e), 0, L(e) - x(e)), 0, 1), \mathsf{ite}(x(e) \leq L(e), 0, 1) \Big) \tag{4}$$

Because the set of index elements E is of arbitrary size and each summand satisfies the same QFPA formula, formula (4) is equisatisfiable with the following formula (5), which uses a different sum operator that computes the set of all sums of solution vectors of the given QFPA formula F:

$$k_1 \neq k_2 - 1 \ \wedge (k_1, k_2, 1, 0, 0) \in \sum_F (x_1, L, x, z_1, z_2), \tag{5}$$

Here F is $z_1 = \mathsf{ite}(x_1 = \mathsf{ite}(L \leq x, 0, L - x), 0, 1) \wedge z_2 = \mathsf{ite}(x \leq L, 0, 1)$. We next show that $(u_1, \ldots, u_n) \in \sum_F (x_1, \ldots, x_c)$ can be replaced with the equisatisfiable QFPA formula. This will reduce the entire problem to QFPA satisfiability.

We first characterize satisfying assignments for F using semilinear sets [8]. This construction is always possible, as described in Section 5. The satisfying assignments for our formula F are given by $\bigcup_{i=1}^{7} (A_i + B_i^*)$ where

$$\begin{array}{l}
A_1 = \{(0,0,0,0,0)\}, B_1 = \{(0,1,1,0,0)\} \\
A_2 = \{(1,0,0,1,0)\}, B_2 = \{(0,1,1,0,0), (1,0,0,0,0)\} \\
A_3 = \{(0,0,1,0,1)\}, B_3 = \{(0,1,1,0,0), (0,0,1,0,0)\} \\
A_4 = \{(1,0,1,1,1)\}, B_4 = \{(0,1,1,0,0), (0,0,1,0,0)\} \\
A_5 = \{(1,1,0,0,0)\}, B_5 = \{(1,1,0,0,0), (0,1,1,0,0)\} \\
A_6 = \{(2,1,0,1,0)\}, B_6 = \{(1,1,0,0,0), (0,1,1,0,0), (1,0,0,0,0)\} \\
A_7 = \{(0,0,1,1,0)\}, B_7 = \{(1,1,0,0,0), (0,1,0,0,0), (0,0,1,1,0)\}
\end{array}$$

Here $A + B^*$ denotes the set of sums with exactly one element from A and any number of elements from B.

The meaning of the sum expression then reduces to the condition $(k_1, k_2, 1, 0, 0) \in (\bigcup_{i=1}^{7} (A_i + B_i^*))^*$. In general, this condition is definable using a QFPA formula that uses the finite vectors from A_i and B_i. In our particular case, $(k_1, k_2, 1, 0, 0)$ can only be a linear combination of elements from $A_1 + B_1^*$ and $A_5 + B_5^*$. Such considerations ultimately result in formula $(k_1, k_2) = (\lambda, \lambda + 1)$, so the overall constraint becomes

$$k_1 \neq k_2 - 1 \ \wedge (k_1, k_2) = (\lambda, \lambda + 1) \tag{6}$$

Because (2) and (6) are equisatisfiable and (6) is unsatisfiable, we conclude that (1) is a valid formula.

3 Multiset Constraints

Figure 1 defines constraints whose satisfiability we study in this paper. Our constraints combine multiset expressions and two kinds of QFPA formulas: *outer linear arithmetic formulas*, denoting relationship between top-level integer values in the constraint, and *inner linear arithmetic formulas*, denoting constraints specific to a given index element $e \in E$. Note that the syntax is not minimal; we subsequently show how many of the constructs are reducible to others.

top-level formulas:

$F ::= A \mid F \wedge F \mid \neg F$

$A ::= M{=}M \mid M \subseteq M \mid \forall e.\mathsf{F}^{\mathsf{in}} \mid \mathsf{A}^{\mathsf{out}}$

outer linear arithmetic formulas:

$\mathsf{F}^{\mathsf{out}} ::= \mathsf{A}^{\mathsf{out}} \mid \mathsf{F}^{\mathsf{out}} \wedge \mathsf{F}^{\mathsf{out}} \mid \neg\mathsf{F}^{\mathsf{out}}$

$\mathsf{A}^{\mathsf{out}} ::= \mathsf{t}^{\mathsf{out}} \leq \mathsf{t}^{\mathsf{out}} \mid \mathsf{t}^{\mathsf{out}}{=}\mathsf{t}^{\mathsf{out}} \mid (\mathsf{t}^{\mathsf{out}}, \ldots, \mathsf{t}^{\mathsf{out}}){=}\sum_{\mathsf{F}^{\mathsf{in}}}(\mathsf{t}^{\mathsf{in}}, \ldots, \mathsf{t}^{\mathsf{in}})$

$\mathsf{t}^{\mathsf{out}} ::= k \mid \; |M| \; \mid C \mid \mathsf{t}^{\mathsf{out}} + \mathsf{t}^{\mathsf{out}} \mid C \cdot \mathsf{t}^{\mathsf{out}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}})$

inner linear arithmetic formulas:

$\mathsf{F}^{\mathsf{in}} ::= \mathsf{A}^{\mathsf{in}} \mid \mathsf{F}^{\mathsf{in}} \wedge \mathsf{F}^{\mathsf{in}} \mid \neg\mathsf{F}^{\mathsf{in}}$

$\mathsf{A}^{\mathsf{in}} ::= \mathsf{t}^{\mathsf{in}} \leq \mathsf{t}^{\mathsf{in}} \mid \mathsf{t}^{\mathsf{in}}{=}\mathsf{t}^{\mathsf{in}}$

$\mathsf{t}^{\mathsf{in}} ::= m(e) \mid C \mid \mathsf{t}^{\mathsf{in}} + \mathsf{t}^{\mathsf{in}} \mid C \cdot \mathsf{t}^{\mathsf{in}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}})$

multiset expressions:

$M ::= m \mid \emptyset \mid M \cap M \mid M \cup M \mid M \uplus M \mid M \setminus M \mid M \setminus\setminus M \mid \mathsf{setof}(M)$

terminals:

m - multiset variables; e - index variable (fixed)

k - integer variable; C - integer constant

Fig. 1. Quantifier-Free Multiset Constraints with Cardinality Operator

Formulas (F) are propositional combinations of atomic formulas (A). Atomic formulas can be multiset equality and subset, pointwise linear arithmetic constraint $\forall e.\mathsf{F}^{\mathsf{in}}$, or atomic outer linear arithmetic formulas ($\mathsf{A}^{\mathsf{out}}$). Outer linear arithmetic formulas are equalities and inequalities between outer linear arithmetic terms ($\mathsf{t}^{\mathsf{out}}$), as well as summation constraints of the form $(u_1, \ldots, u_n) = \sum_F (t_1, \ldots, t_n)$, which compute the sum of the vector expression $(t_1, \ldots, t_n)$ over all indices $e \in E$ that satisfy the formula F. Outer linear arithmetic terms ($\mathsf{t}^{\mathsf{out}}$) are built using standard linear arithmetic operations starting from: 1) integer variables (k), 2) cardinality expressions applied to multisets ($|M|$), and 3) integer constants (C). The $\mathsf{ite}(F, t_1, t_2)$ expression is the standard if-then-else construct, whose value is t_1 when F is true and t_2 otherwise. Inner linear arithmetic formulas are linear arithmetic formulas built starting from constants (C) and values $m(e)$ of multiset variables at the current index e.

Multiset constraints contain some common multiset operations such as disjoint union, intersection, and difference, as well as the setof operation that computes the largest set contained in a given multiset. Additionally, using the constraints $\forall e.\mathsf{F}^{\mathsf{in}}$ it is possible to specify any multiset operation defined pointwise using a QFPA formula. Note also that it is easy to reason about individual elements of sets at the top level by representing them as multisets s such that $|s| = 1$. If s is such a multiset representing an element and m is a multiset, we can count the number of occurrences of s in m with, for example, the expression $\sum \mathsf{ite}(s(e){=}0, 0, m(e))$.

4 Reducing Multiset Operations to Sums

We next show that all operations and relations on multisets as a whole can be eliminated from the language of Figure 1. To treat operations as relations, we flatten formulas by introducing fresh variables for subterms and using the equality operator. Figure 2 summarizes this process.

Definition 1 (Sum normal form). *A multiset formula is in* sum normal form *iff it is of the form $P \wedge (u_1, \ldots, u_n) = \sum_{\mathit{true}} (t_1, \ldots, t_n)$, where P is a quantifier-free Presburger arithmetic formula without any multiset variables, and the variables in $t_1, \ldots, t_n$ occur only as expressions of the form $m(e)$ for m a multiset variable and e the fixed index variable.*

Theorem 1 (Reduction to sum normal form). *Algorithm in Figure 2 reduces in polynomial time any formula in the language of Figure 1 to a formula in sum normal form.*

4.1 From Multisets to Sum Constraints

We next argue that formulas in sum normal form (Definition 1) are equisatisfiable with formulas of linear arithmetic extended with sum constraints (Figure 3). Sum constraints are of the form $(u_1, \ldots, u_n) {\in} \sum_F (t_1, \ldots, t_n)$ and they test membership in the set of vectors generated using vector addition starting from the set $\{(t_1, \ldots, t_n) \mid \exists k_1, \ldots, k_n . F\}$ where $k_1, \ldots, k_n$ is the set of all variables occurring in F but not occurring in $t_1, \ldots, t_n$.

Theorem 2 (Multiset elimination). *Consider a sum normal form formula F of the form*

$$P \ \wedge \ (u_1, \ldots, u_n) = \sum_{\mathit{true}} (t_1, \ldots, t_n)$$

where free variables of $t_1, \ldots, t_n$ are multiset variables $m_1, \ldots, m_q$. Let $k_1, \ldots, k_q$ be fresh integer variables. Then F is equisatisfiable with the formula

$$P \ \wedge \ (u_1, \ldots, u_n) \in \sum_{\mathit{true}} (t'_1, \ldots, t'_n) \tag{7}$$

where $t'_i = t_i[m_1(e) := k_1, \ldots, m_q(e) := k_q]$ (t'_i results from t_i by replacing multiset variables with fresh integer variables).

The equisatisfiability follows by bijection between the satisfying assignments where k_i is interpreted as $m_i(e)$ and E has as many elements as there are summands under the sum in (7).

INPUT: multiset formula in the syntax of Figure 1
OUTPUT: formula in sum-normal form (Definition 1)

1. Flatten expressions that we wish to eliminate:
   $C[e] \rightsquigarrow (x = e \wedge C[x])$
   where e is one of the expressions $\emptyset$, $m_1 \cup m_2$, $m_1 \cup m_2$, $m_1 \uplus m_2$, $m_1 \setminus m_2$, $\mathsf{setof}(m_1)$, $|m_1|$, and where the occurrence of e is not already in a top-level conjunct of the form $x = e$ or $e = x$ for some variable x.
2. Reduce multiset relations to pointwise linear arithmetic conditions:
   $$\begin{array}{lcl}
   C[m_0 = \emptyset] & \rightsquigarrow & C[\forall e.\ m_0(e) = 0] \\
   C[m_0 = m_1 \cap m_2] & \rightsquigarrow & C[\forall e.\ m_0(e) = \mathsf{ite}(m_1(e) \leq m_2(e), m_1(e), m_2(e))] \\
   C[m_0 = m_1 \cup m_2] & \rightsquigarrow & C[\forall e.\ m_0(e) = \mathsf{ite}(m_1(e) \leq m_2(e), m_2(e), m_1(e))] \\
   C[m_0 = m_1 \uplus m_2] & \rightsquigarrow & C[\forall e.\ m_0(e) = m_1(e) + m_2(e)] \\
   C[m_0 = m_1 \setminus m_2] & \rightsquigarrow & C[\forall e.\ m_0(e) = \mathsf{ite}(m_1(e) \leq m_2(e), 0, m_1(e) - m_2(e))] \\
   C[m_0 = m_1 \setminus\setminus m_2] & \rightsquigarrow & C[\forall e.\ m_0(e) = \mathsf{ite}(m_2(e) = 0, m_1(e), 0)] \\
   C[m_0 = \mathsf{setof}(m_1)] & \rightsquigarrow & C[\forall e.\ m_0(e) = \mathsf{ite}(1 \leq m_1(e), 1, 0)] \\
   C[m_1 \subseteq m_2] & \rightsquigarrow & C[\forall e.\ (m_1(e) \leq m_2(e))] \\
   C[m_1 = m_2] & \rightsquigarrow & C[\forall e.\ (m_1(e) = m_2(e))]
   \end{array}$$
3. Express each pointwise constraint using a sum:
   $C[\forall e.F] \rightsquigarrow C[\sum_{\neg F} 1 = 0]$
4. Express each cardinality operator using a sum:
   $C[\,|m|\,] \rightsquigarrow C[\sum_{\mathsf{true}} m(e)]$
5. Flatten any sums that are not already top-level conjuncts:
   $$C[\,(u_1, \ldots, u_n) {=} \sum_F (t_1, \ldots, t_n)\,] \;\rightsquigarrow\; (w_1, \ldots, w_n) {=} \sum_F (t_1, \ldots, t_n) \wedge C[\,\bigwedge_{i=1}^{n} u_i {=} w_i\,]$$
6. Eliminate conditions from sums:
   $C[\,\sum_F (t_1, \ldots, t_n)\,] \rightsquigarrow C[\,\sum_{\mathsf{true}} (\mathsf{ite}(F, t_1, 0), \ldots, \mathsf{ite}(F, t_n, 0))\,]$
7. Group all sums into one:
   $$P \wedge \bigwedge_{i=1}^{q} (u_1^i, \ldots, u_{n_i}^i) = \sum_{\mathsf{true}} (t_1^i, \ldots, t_{n_i}^i) \;\rightsquigarrow$$
   $$P \wedge (u_1^1, \ldots, u_{n_1}^1, \ldots, u_1^q, \ldots, u_{n_q}^q) = \sum_{\mathsf{true}} (t_1^1, \ldots, t_{n_1}^1, \ldots, t_1^q, \ldots, t_{n_q}^q)$$

Fig. 2. Algorithm for reducing multiset formulas to sum normal form

5 Deciding Linear Arithmetic with Sum Constraints

Having reduced in polynomial time multiset constraint satisfiability to satisfiability for linear arithmetic with sum constraints, this section examines the decidability and the complexity of the resulting satisfiability problem.

5.1 Preliminary Transformations

We assume that the constraint whose satisfiability we wish to check is in the form given by Theorem 2. This is sufficient for deciding multiset constraints. We

top-level, outer linear arithmetic formulas:
$\mathsf{F}^{\mathsf{out}} ::= \mathsf{A}^{\mathsf{out}} \mid \mathsf{F}^{\mathsf{out}} \wedge \mathsf{F}^{\mathsf{out}} \mid \neg\mathsf{F}^{\mathsf{out}}$
$\mathsf{A}^{\mathsf{out}} ::= \mathsf{t}^{\mathsf{out}} \leq \mathsf{t}^{\mathsf{out}} \mid \mathsf{t}^{\mathsf{out}}{=}\mathsf{t}^{\mathsf{out}} \mid (\mathsf{t}^{\mathsf{out}}, \ldots, \mathsf{t}^{\mathsf{out}}) {\in} \sum_{\mathsf{F}^{\mathsf{in}}} (\mathsf{t}^{\mathsf{in}}, \ldots, \mathsf{t}^{\mathsf{in}})$
$\mathsf{t}^{\mathsf{out}} ::= \mathsf{k}^{\mathsf{out}} \mid C \mid \mathsf{t}^{\mathsf{out}} + \mathsf{t}^{\mathsf{out}} \mid C \cdot \mathsf{t}^{\mathsf{out}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}})$
inner linear arithmetic formulas:
$\mathsf{F}^{\mathsf{in}} ::= \mathsf{A}^{\mathsf{in}} \mid \mathsf{F}^{\mathsf{in}} \wedge \mathsf{F}^{\mathsf{in}} \mid \neg\mathsf{F}^{\mathsf{in}}$
$\mathsf{A}^{\mathsf{in}} ::= \mathsf{t}^{\mathsf{in}} \leq \mathsf{t}^{\mathsf{in}} \mid \mathsf{t}^{\mathsf{in}}{=}\mathsf{t}^{\mathsf{in}}$
$\mathsf{t}^{\mathsf{in}} ::= \mathsf{k}^{\mathsf{in}} \mid C \mid \mathsf{t}^{\mathsf{in}} + \mathsf{t}^{\mathsf{in}} \mid C \cdot \mathsf{t}^{\mathsf{in}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}})$
terminals:
$\mathsf{k}^{\mathsf{in}}, \mathsf{k}^{\mathsf{out}}$ - integer variable (two disjoint sets); C - integer constants

Fig. 3. Syntax of Linear Arithmetic with Sum Constraints

therefore consider a formula of the form

$$P \;\wedge\; (u_1, \ldots, u_n) \in \sum_{\mathsf{true}} (m_1, \ldots, m_n) \tag{8}$$

Let $x_1, \ldots, x_q$ be the set of variables in $m_1, \ldots, m_n$ and let $y_1, \ldots, y_q$ and $z_1, \ldots, z_n$ be fresh variables. We then represent (8) as the formula $P \wedge S$ where S is the formula

$$(u_1, \ldots, u_n, y_1, \ldots, y_q) \in \sum_{F} (z_1, \ldots, z_n, x_1, \ldots, x_q) \tag{9}$$

Here F is the formula $\bigwedge_{i=1}^{q} m_i = z_i$. (The values $y_1, \ldots, y_q$ are not used in P; their purpose is to ensure proper dimensionality of the resulting vector, so that we can assume that all variables of F appear in vector $(z_1, \ldots, z_n, x_1, \ldots, x_q)$.) Note that the formula S says that some number of solutions of QFPA formula F sums up to a given vector. We next show that S is equisatisfiable with a QFPA formula.

5.2 Formula Solutions as Semilinear Sets and their Bounds

To show that QFPA formulas are closed under unbounded sum constraints, we use representations of solutions of QFPA formulas as semilinear sets. We first review some relevant results from [23]. For an integer vector $x = (x^1, \ldots, x^n)$ let $||x||_1$ denote $\sum_{i=1}^{n} |x_i|$. For a matrix $A = [a_{ij}]$ let $||A||_{1,\infty}$ denote $\sup_i(\sum_j a_{ij})$.

Definition 2 (Sum and Iteration of Sets of Vectors). *Let $C_1, C_2 \subseteq \mathbb{N}^k$ be sets of vectors of non-negative integers. We define*

$$\begin{aligned} C_1 + C_2 &= \{x_1 + x_2 \mid x_1 \in C_1 \wedge x_2 \in C_2\} \\ C_1^* &= \{0\} \cup \{x_1 + \ldots + x_n \mid x_1, \ldots, x_n \in C_1\} \end{aligned}$$

When $x \in \mathbb{N}^n$ and $C_2 \subseteq \mathbb{N}^n$ is finite, we call $\{x\} + C_2^*$ a linear set*. A* semilinear set *is a union of some finite number of linear sets.*

If $C_1, C_2 \subseteq \mathbb{N}^n$ are finite, then $C_1 + C_2^*$ is a particular kind of a semilinear set. Such semilinear sets are solutions of systems of linear equations, which follows from the proof of [23, Corollary 1] and [23, Theorem 1]:

Fact 1 (Pottier 1991) *Consider a system of equations $Ax = b$ where $A \in \mathbb{N}^{m,n}$ and $b \in \mathbb{N}^m$. Let $A_1 = [A; -b]$, let r be the rank of A_1, and let $B_0 = (1 + ||A_1||_{1,\infty})^r$. Then there exist two finite sets $C_1, C_2 \subseteq \mathbb{N}^n$ such that*

1. *for all $x \in \mathbb{N}^n$, $Ax = b$ iff $x \in C_1 + C_2^*$, and*
2. *$\forall h \in C_1 \cup C_2$, $||h||_1 \leq B_0$.*

Consequently, $|C_1| \leq B_0$ and $|C_2| \leq B_0$. Moreover, if in $Ax = b$ we replace some of the equations with inequations, the statement still holds if B_0 is weakened to $(2 + ||A_1||_{1,\infty})^m$.

Note that each QFPA formula F can be converted into an equivalent disjunction of systems of equations and inequations. The number of such systems is singly exponential in the number of atomic formulas in F. Moreover, the elements of A and b in the resulting systems are polynomially bounded by the coefficients and constants in the original QFPA formula. Consequently, the B_0 bound for each of these systems is at most singly exponential in the size s of the formula F. We denote this bound by $2^{p(s)}$, where p is some polynomial that follows from details of the algorithm for generating all systems of equations and inequations whose disjunction is equivalent to F. We thus obtain the following lemma.

Lemma 1. *Let F be a QFPA formula of size s with n free variables. Then there exist finite sets $A_i, B_i \subseteq \mathbb{N}^n$ for $1 \leq i \leq d$ for some $d \leq 2^{p_1(s)}$ such that the set of satisfying assignments for F is given as*

$$\bigcup_{i=1}^{d} (A_i + B_i^*)$$

and such that $||h||_1 \leq 2^{p(s)}$ for each $h \in \bigcup_{i=1}^{d} (A_i \cup B_i)$.

If $A = \{a_1, \ldots, a_q\}$ and $B = \{b_1, \ldots, b_r\}$ for $a_i, b_j \in \mathbb{N}^n$, then the condition $u \in A + B^*$ is given by the formula $\bigvee_{i=1}^{q} (u = a_i + \sum_{j=1}^{r} \lambda_j b_j)$ where $\lambda_1, \ldots, \lambda_r$ are existentially quantified variables ranging over $\mathbb{N}$. This view leads to the following formulation of Lemma 1.

Lemma 2 (Semilinear normal form for linear arithmetic). *Let F be a QFPA formula of size s with n free variables. Then there exist vectors a_i and b_{ij}, $1 \leq j \leq q_i$, $1 \leq i \leq d$ for $d \leq 2^{p_1(s)}$, with $||a_i||_1, ||b_{ij}||_1 \leq 2^{p(s)}$ such that F is equivalent to*

$$\exists \lambda_1, \ldots, \lambda_q. \bigvee_{i=1}^{d} (u = a_i + \sum_{j=1}^{q_i} \lambda_j b_{ij}) \tag{10}$$

where $u = (u_1, \ldots, u_n)$ are the free variables in F and q is the maximum of all q_i.

5.3 Formulas Representing Unbounded Sums

Having obtained semilinear normal form for QFPA formulas, we can characterize the set of all sums of solutions of a formula. This corresponds to taking the set of solutions C and computing a representation of C^*. We next give a QFPA formula for C^* (this was also obtained in [14, Section 3.2]).

Lemma 3. *Given a formula F in normal form (10), if x denotes vector of variables $(x_1, \ldots, x_n)$ then the condition $x \in \sum_F (u_1, \ldots, u_n)$ is equivalent to*

$$\exists \mu_i, \lambda_{ij}.\ x = \sum_{i=1}^{d} (\mu_i a_i + \sum_{j=1}^{q_i} \lambda_{ij} b_{ij}) \ \wedge\ \bigwedge_{i=1}^{d} (\mu_i = 0 \implies \sum_{j=1}^{q_i} \lambda_{ij} = 0) \tag{11}$$

The existentially quantified variables μ_i, λ_{ij} become free variables in the satisfiability problem. We have therefore reduced the original formula $P \wedge S$ where S is given by (8) to conjunction of P and (11), which is a QFPA formula. Along with the algorithm in Figure 2, this shows the decidability of the satisfiability problem for multiset constraints in Figure 1.

5.4 Bounds on Solutions for Formulas with Sums

The algorithm described so far produces exponentially large QFPA formulas, so it would only give a non-deterministic exponential bound on the satisfiability problem. To improve this complexity upper bound, we establish bounds on values of variables $(u_1, \ldots, u_n)$ in (8). As the first step, we rewrite (11) by applying case analysis, for each i, on whether $\mu_i = 0$ or $\mu_i \geq 1$. We obtain the formula

$$\exists \mu_i, \lambda_{ij}. \bigvee_{I \subseteq \{1,\ldots,d\}} x = \sum_{i \in I} ((1 + \mu_i) a_i + \sum_{j=1}^{q_i} \lambda_{ij} b_{ij}) \tag{12}$$

The key property of (12) is that, although it can still have exponentially large number of variables in the size of the original formula S, each of the disjuncts in disjunctive normal form of (12) has a polynomial number of atomic formulas. In other words, the formula can be represented as a disjunction of systems of equations whose matrices A have polynomially many rows (and potentially exponentially many columns). Consequently, the same property holds for the conjunction $P \wedge$(12). This allows us to proceed similarly as in [15, Section 3]. We apply the well-known bound on integer linear programming problems.

Fact 2 (Papadimitriou [21]) *Let A be an $m \times n$ integer matrix and b an m-vector, both with entries from $[-a..a]$. Then the system $Ax = b$ has a solution in $\mathbb{N}^m$ if and only if it has a solution in $[0..M]^m$ where $M = n(ma)^{2m+1}$.*

Given that all coefficients appearing in (12) are bounded by $2^{p(s)}$ and that m is polynomial in s as well, we obtain the desired bound.

Theorem 3. *There exists a polynomial $p(s)$ such that for every formula F of Figure 3 of the form (8) and of size s, F has a solution iff F has a solution in which the number of bits needed to represent the values of outer integer variables is bounded by $p(s)$.*

By Theorem 3 there is a non-deterministic polynomial time algorithm that

1. guesses the values $c_1, \ldots, c_{n+q}$ of variables $u_1, \ldots, u_n, y_1, \ldots, y_q$ in (9) such that P holds, and then
2. checks whether the constraint $(c_1, \ldots, c_{n+q}) \in \sum_F(z_1, \ldots, z_n, x_1, \ldots, x_q)$ has a solution.

We have therefore reduced the satisfiability problem to testing whether a given vector of non-negative integers is a sum of some number of solutions of F. This test is the subject of the next section.

5.5 PSPACE Algorithm for Sum Membership

This section examines the problem of checking for a given constant vector $\boldsymbol{c} \in \mathbb{N}$ and a given QFPA formula F, whether $\boldsymbol{c} \in \{\boldsymbol{v} \mid F(\boldsymbol{v})\}^*$ holds, that is, whether there exists some number $q \geq 0$ of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_q \in \mathbb{N}^n$ such that $\sum_{i=1}^{q} \boldsymbol{v}_i = \boldsymbol{c}$ and $F(\boldsymbol{c})$ holds forall $1 \leq i \leq q$. In principle, this problem could be solved by checking the satisfiability of formula (11). However, the number and size of vectors a_i and b_{ij} is exponential. The techniques that we know for constructing them are based on computing Hilbert basis of homogeneous systems of equations over natural numbers ($Ax = 0$) [23,4]. In [5] the authors show that counting the number of solutions of Hilbert basis of a system of equations is complete for the counting class #coNP.

We therefore adopt a more direct approach to checking $\boldsymbol{c} \in \{\boldsymbol{v} \mid F(\boldsymbol{v})\}^*$, which does not attempt to compute semilinear sets for F. A simple non-deterministic polynomial-space algorithm would guess non-zero solutions of the formula F that are bounded by $\boldsymbol{c}$ and subtract them from $\boldsymbol{c}$ until it reaches the zero vector. Figure 4 we presents a refinement of this algorithm that uses divide and conquer approach and can easily be implemented deterministically in polynomial space. Note that in invocations of depth up to t the algorithm will find sums $\boldsymbol{v}_1 + \ldots + \boldsymbol{v}_q = \boldsymbol{c}$ for all $q \leq 2^t$. Because the coordinates of solutions are non-negative integers, it suffices to consider sums of length up to $||c||_1$, which is bounded by $2^{p(s)}$. Therefore, the bound $p(s)$ on the depth of recursion suffices.

The algorithm in Figure 4 also gives a natural encoding of the problem into Presburger arithmetic with bounded quantifiers, which is PSPACE complete. Namely, we can rewrite the two recursive calls in $\mathsf{generated}(\boldsymbol{c}_1, t-1) \wedge \mathsf{generated}(\boldsymbol{c}_2, t-1)$ as

$$\forall \boldsymbol{a}.\ (\boldsymbol{a} = \boldsymbol{c}_1 \vee \boldsymbol{a} = \boldsymbol{c}_2) \implies \mathsf{generated}(\boldsymbol{a}, t-1) \tag{13}$$

Given a formula F of size s, we then unroll the recursion $p(s)$ times, which eliminates all recursive calls and the parameter t. Because (13) contains only

INPUT: A vector $\boldsymbol{c} \in \mathbb{N}^n$, and a QFPA formula F of size s with free variables $\boldsymbol{k} = (k^1, \ldots, k^n)$.
OUTPUT: true iff $\boldsymbol{c} \in \{\boldsymbol{v} \mid F(\boldsymbol{v})\}^*$
TOP LEVEL: return generated$(\boldsymbol{c}, p(s))$;

```
proc generated(c, t) :
  if(c = 0) then return true;
  if(F(c) = true) then return true;
  if(t = 0) then return false;
  non-deterministically guess c1, c2 ∈ N^n \ {0} such that c1 + c2 = c;
  return (generated(c1, t − 1) ∧ generated(c2, t − 1));
```

Fig. 4. PSPACE Algorithm for testing whether a vector is a sum of solutions of a QFPA formula

one recursive call, the resulting unrolling is polynomially large and it can be encoded as a QFPA formula. This formula contains universal quantification over the $\boldsymbol{a}$ vectors and existential quantifiers over the $\boldsymbol{c}_1, \boldsymbol{c}_2$ vectors in each step of the recursion. It has polynomial size and $p(s)$ quantifier alternations.

Theorem 4 (Membership test). *The algorithm in Figure 4 is correct and runs in polynomial space.*

From Theorem 1, Theorem 2, and Theorem 4, we obtain our main result.

Corollary 1. *The satisfiability problem for the language in Figure 1 is decidable and belongs to PSPACE.*

6 Undecidability of Quantified Constraints

We next show that adding quantifiers to the language of Figure 1 (and to many of its fragments) results in undecidable constraints.

The language in Figure 1 can be seen as a generalization of quantifier-free Boolean algebra with Presburger arithmetic (QFBAPA) [12]. Given that QFBAPA admits quantifier elimination [7,10], it is interesting to ask whether multiset quantifiers can be eliminated from constraints of the present paper. Note that a multiset structure without cardinality operator can be viewed as a product of Presburger arithmetic structures. Therefore, Feferman-Vaught theorem [7] (see also [3], [11, Section 3.3]) gives a way to decide the first-order theory of multiset operations extended with the ability to state cardinality of sets of the form $|\{e \mid F(e)\}|$. This corresponds to multiset theory with counting distinct elements of multisets, which is denoted $FO^{\#D}_{\mathcal{M}}$ in [13]. However, this language is strictly less expressive than a quantified extension of the language in Figure 1 that contains summation expressions $\sum_{F(e)} t(e)$ and that corresponds to $FO^{\#}_{\mathcal{M}}$ in [13]. The decidability of $FO^{\#}_{\mathcal{M}}$ was stated as open in [13]. We next show that this language is undecidable.

The undecidability follows by reduction from Hilbert's 10th problem [16], because quantified multiset constraints can define not only addition (using disjoint union $\uplus$) but also multiplication. To define $x \cdot y = z$, we introduce a new set p that contains x distinct elements, each of which occurs y times. The following formula encodes this property.

$$x \cdot y = z \Leftrightarrow \exists p.\ z = |p| \ \wedge\ x = |\mathsf{setof}(p)| \ \wedge \\ (\forall m.\ |m| = z \wedge |\mathsf{setof}(m)| = 1 \wedge \mathsf{setof}(m) \subseteq p \implies |m \cap p| = y)$$

Because we can define multiplication, we can express satisfiability of Diophantine equations, so by [16] we conclude that satisfiability of multiset constraints with quantifiers and cardinality is undecidable. Similarly, we obtain undecidable constraints if in the quantified expressions $\forall e.F$ we admit the use of outer integer variables as parameters. This justifies the current "stratified" syntax that distinguishes inner and outer integer variables.

The reader may wonder whether the presence of the built-in setof operator is needed for undecidability of quantified constraints. However, the setof operator is itself definable using quantifiers. For example, $a = \mathsf{setof}(b)$ iff a is the smallest set that behaves the same as b with respect to simple set membership. Behaving same with respect to simple set membership is given by

$$\mathsf{memSame}(a, b) \iff (\forall x. |x| = 1 \implies (x \subseteq a \iff x \subseteq b))$$

so $a = \mathsf{setof}(b) \iff (\mathsf{memSame}(a, b) \wedge (\forall a_1.\ \mathsf{memSame}(a_1, b) \implies a \subseteq a_1))$. Moreover, note that, as in any lattice, $\cap$ and $\subseteq$ are inter-expressible using quantifiers. Therefore, adding quantifiers to a multiset language that contains $\subseteq$ and cardinality constructs already gives undecidable constraints. This answers negatively the question on decidability of $FO^{\#D}_{\mathcal{M}}$ posed in [13, Section 3.4].

7 Conclusions

Motivated by applications in verification, we introduced an expressive class of constraints on multisets. Our constraints support arbitrary multiset operations defined pointwise using QFPA as well as the cardinality operator. We presented a decision procedure for the satisfiability of these constraints, showing that they efficiently reduce to an extension of QFPA with unbounded sum expressions. For the later problem we presented a decision procedure based on semilinear set representation of quantifier-free Presburger arithmetic formulas. We established small bounds on solutions of such formulas and then showed that the overall problem can be solved in polynomial space. The satisfiability problem for our constraints is therefore NP-hard and belongs to PSPACE.[1] Finally, we showed that adding quantifiers to these constraints makes them undecidable by defining multiplication in the language.

[1] We recently established NP-completeness of these constraints by generalizing [6].

References

1. Alexander Aiken. Introduction to set constraint-based program analysis. *Science of Computer Programming*, 35:79–111, 1999.
2. Jean-Pierre Banâtre and Daniel Le Métayer. Programming by multiset transformation. *Commun. ACM*, 36(1):98–111, 1993.
3. Alexis Bès. Definability and decidability results related to the elementary theory of ordinal multiplication. *Fund.Math*, 171:197–211, 2002.
4. Eric Domenjoud. Solving systems of linear diophantine equations: An algebraic approach. In *MFCS*, pages 141–150, 1991.
5. Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. *Theor. Comput. Sci.*, 340(3):496–513, 2005.
6. Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, September 2006.
7. S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.
8. S. Ginsburg and E. Spanier. Semigroups, Pressburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
9. Viktor Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.
10. Viktor Kuncak, Hai Huu Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *J. of Automated Reasoning*, 2006.
11. Viktor Kuncak and Martin Rinard. On the theory of structural subtyping. Technical Report 879, LCS, Massachusetts Institute of Technology, 2003.
12. Viktor Kuncak and Martin Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In *CADE-21*, 2007.
13. D. Lugiez. Multitree automata that count. *Theor. Comput. Sci.*, 333(1-2):225–263, 2005.
14. Denis Lugiez and Silvano Dal Zilio. Multitrees Automata, Presburger's Constraints and Tree Logics. Research report 08-2002, LIF, Marseille, France, June 2002. http://www.lif-sud.univ-mrs.fr/Rapports/08-2002.html.
15. Bruno Marnette, Viktor Kuncak, and Martin Rinard. On algorithms and complexity for sets with cardinality constraints. Technical report, MIT CSAIL, August 2005.
16. Yuri V. Matiyasevich. Enumerable sets are Diophantine. *Soviet Math. Doklady*, 11(2):354–357, 1970.
17. Jayadev Misra. A logic for concurrent programming (in two parts): Safety and progress. *Journal of Computer and Software Engineering*, 3(2):239–300, 1995.
18. Huu Hai Nguyen, Cristina David, Shengchao Qin, and Wei-Ngan Chin. Automated verification of shape, size and bag properties via separation logic. In *VMCAI*, 2007.
19. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.
20. Tobias Nipkow, Markus Wenzel, Lawrence C Paulson, and Norbert Voelker. Multiset theory version 1.30 (Isabelle distribution). http://isabelle.in.tum.de/dist/library/HOL/Library/Multiset.html, 2005.
21. Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

22. Lawrence C. Paulson. Mechanizing a theory of program composition for UNITY. *ACM Trans. Program. Lang. Syst.*, 23(5):626–656, 2001.
23. Loïc Pottier. Minimal solutions of linear diophantine systems: Bounds and algorithms. In *RTA*, volume 488 of *LNCS*, 1991.
24. J. T. Schwartz. On programming: An interim report on the SETL project. Technical report, Courant Institute, New York, 1973.
25. Calogero G. Zarba. Combining multisets with integers. In *CADE-18*, 2002.